What is claimed is:

1. A method comprising:

executing a first instruction in a processor;

if the execution of the first instruction generates a cache miss, associating the first instruction with the cache miss;

enqueuing the first instruction for re-execution; and

after the cache miss with which the first instruction is associated is serviced, re-executing the first instruction.

2. The method of claim 1, further comprising associating the cache miss with a second instruction dependent on the first instruction.

3. The method of claim 1, further comprising assigning an identifier to the cache miss.

4. The method of claim 1, further comprising determining a priority of the instruction.

5. A processor comprising:

a re-scheduler to hold instructions enqueued for execution; and

association logic to form an association between a cache miss and an instruction generating the cache miss, the instruction to be enqueued in the re-scheduler.

6. The processor of claim 5, wherein the re-scheduler is further coupled to priority logic to determine a priority of instructions in the re-scheduler.

7. The processor of claim 5, wherein the association logic is to assign an identifier to the cache miss.

8. The processor of claim 5, wherein the re-scheduler is to receive a signal indicating that the cache miss corresponding to the association has been serviced.

9. The processor of claim 8, wherein the re-scheduler is to cause an instruction to be designated as eligible for re-execution based on the signal.

10. A method comprising:

generating a cache miss in a processor;

assigning an identifier to the cache miss and writing the identifier in a field of a load instruction generating the cache miss;

issuing a request to service the cache miss to a memory system of the computer and providing the identifier to the memory system;

placing the load instruction in a queue for re-execution, where an eligibility of the instruction for re-execution is based at least in part on the identifier;

after the memory system completes servicing the request, causing the memory system to provide the identifier to the queue; and

designating the load instruction as eligible for re-execution based on the identifier provided by the memory system.

11. The method of claim 10, further comprising re-executing the load instruction based on receiving the identifier from the memory system.

12. The method of claim 10, further comprising propagating the identifier to any instruction dependent on the load instruction.

13. A method comprising:

in a processor, enqueuing a plurality of instructions needing re-execution due to respective cache misses in a re-execution queue;

associating each instruction in the queue with a respective corresponding cache miss; and

after a cache miss is serviced, re-executing those instructions in the re-execution queue associated with the serviced cache miss.

14. The method of claim 13, further comprising determining a priority of the instructions.

15. The method of claim 13, wherein the associating comprises writing an identifier of a cache miss in an instruction.

16. A system comprising:

a memory system to hold instructions for execution;

a processor coupled to the memory system, the processor including:

a re-scheduler to hold instructions from the memory system enqueued for execution; and

association logic to form an association between a cache miss and an instruction generating the cache miss, the instruction to be enqueued in the re-scheduler.

17. The system of claim 16, wherein the re-scheduler is further coupled to priority logic to determine a priority of instructions in the re-scheduler.

18. The system of claim 16, wherein the association logic is to assign an identifier to the cache miss.